**ns models**

- Wired (point-to-point, LANs, wireless (multiple)
- Physical model
- Link-level protocols, e.g. Ethernet, FDDI, etc.
- Wide-area routing, e.g. BGP-4 and directed diffusion
- Routing and queueing

**Traffic models and applications**

- Web, FTP, Voice, real-time audio, etc.
- Traffic models, independent of model

**ns goals**

- Multiple levels of detail: from ns to real
- Experts develop models
- Many protocol test models in various simulations
- Increase confidence in results
- Provide a collaborative environment
- Provide a source of test models, etc.
- Support networking research and education

**ns history**

- Current maintenance at USC/ISI with input from Floyd et al.
- ns-2 by McCanne and VINT project (LBL, PARC, UCB, USC/ISI)
- Began as REAL in 1989

**ns models**

- Transport protocols:
  - TCP (reno, vegas, etc), UDP
- Internet: TCP, IP, ICMP, etc.
- Application: Web, FTP, Voice, real-time audio, etc.

**Traffic models and applications**

- Simple traffic models: Poisson, uniform, etc.
- Traffic and topology generators
- Simple network scenarios
- Visualization of network output
- ns, the network animator

**ns components**

- "ns" components
- "ns" components
- "ns" components

- Traffic and topology generators
- Simple network scenarios
- Visualization of network output
- ns, the network animator

- Protocol design, traffic studies, etc.
- Protocol comparison
- Free and freely distributed, open source
- Support networking research and education
Platforms: basically all Unix and Windows

7

ns status

Computer Communications - CSCI 551
Copyright © William C. Cheng

size: about 200k loc each C++ and Tcl, 350 page manual

user-base: >1k institutions, >10k users

releases about every 6 months, plus daily snapshots

ns Status

Copyright © William C. Cheng

Outlines

Concepts

Getting Started

Fundamental (c.f. p. 13) on

Essentials

Platforms: basically all Unix and Windows

Copyright © William C. Cheng
C++ Software Structure: C++ and Otcl

Pros:
- Trade-off running vs. writing speed
- Powerful/documented config language

Cons:
- Two languages to learn and debug

Otcl for control
- Fast to write and change
- Fast to compile and debug
- Simple configuration
- C++ for object-oriented, complex control

Otcl and C++: The Duality

Hello World, Deconstructed

set ns [new Simulator]
ns at 1.0 "puts "Hello World!"
ns at 1.5 "exit"
ns run

ns allinone (if you have your own machine, do not build on USC servers)

Outlines
- Installation and Documentation
- Getting Started
- Concepts
- Essentials
- Essentials
- Fundamentals

TclCL is glue library that makes it easy to share functions, variables, etc.

ns-utils@isi.edu
http://www.isi.edu/nsnam/ns/

nunki 74%
~csci551/ns simple.tcl
Hello World!

nunki 75%
Simulator *ns=
set ns [new Simulator];
ns->at(1, "puts "Hello World!");
ns->at(1.5, "exit");
ns->run();

Think C++:
- Create a simulator and put in var ns
- Schedule an event at time t=1 to print HW
- Create a simulation 'plan' in var ns
- Set up new simulation

Hello World

Copyright © William C. Cheng
Basic Tcl

- Creating Event Scheduler
- Schedule event
  \[\text{ns at <time> <event>}\]
- Start scheduler
  \[\text{ns run}\]

Nodes

- Creating Network
- Setting up routes
- Creating transport connection
- Setting up nodes

Creating the event scheduler

Basic ns-2

Creating Network

- Creating the event scheduler

Basic otcl

Class Person

# constructor:
Person instproc init {age} {
  $self instvar age_
  set age_ $age
}

# method:
Person instproc greet {} {
  $self instvar age_
  puts "$age_ years old: How are you doing?"
}

# subclass:
Class Kid - superclass Person
Kid instproc greet {} {
  $self instvar age_
  puts "$age_ years old kid: What's up, dude?"
}

set a [new Person 45]
set b [new Kid 15]
$a greet
$b greet

Can easily make variations of existing things (TCP, TCP/Reno)

Creating base classes:

ns -nohyphens
ns at 0 start
ns run

Creating the Event Scheduler

Basic Tcl

Creating the Event Scheduler

Class Person

# constructor:
Person instproc init {age} {
  $self instvar age_
  set age_ $age
}

# method:
Person instproc greet {} {
  $self instvar age_
  puts "$age_ years old: How are you doing?"
}

# subclass:
Class Kid - superclass Person
Kid instproc greet {} {
  $self instvar age_
  puts "$age_ years old kid: What's up, dude?"
}

set a [new Person 45]
set b [new Kid 15]
$a greet
$b greet

Can easily make variations of existing things (TCP, TCP/Reno)
Computing routes

**Unicast**

- Static
- Session
- Dynamic-Vector
- Cost
- Multi-path

**Multicast**

- Controlled-Resource Multicast (CRM)
- Source-Redundancy Multicast (SRM)
- Game-Redundancy Multicast (GRM)
- Zero's Message Multicast (ZMM)

Simple two layers: transport and app

Traffic

Creating Traffic: On Top of TCP

FTP

Creating Connection: TCP

Creating Traffic: On Top of UDP

Creating Traffic: On Top of UDP

Creating Traffic: On Top of TCP
Creating Traffic: Trace Driven

**Computer Communications - CSCI 551**  
Copyright © William C. Cheng

```
set tfile [new Tracefile]
```

Creating Error Module

```
loss_module [new ErrorModel]\n
loss_module set rate_ 0.01\nloss_module unit pkt\nloss_module ranvar [new RandomVariable/Uniform]\nloss_module drop-target [new Agent/Null]
```

Inserting Error Module

```
ns lossmodel\nns test.nam\n```

Trace packets on all links into `test.out`

```
[open test.out w]\n```

Trace packets on all links in nam-1 format

```
[open test.nam w]\n```

**Plumbing: Packet Flow**

```
| 0 1 0 1 0 1 \n-----------------\n\n```

Comparing to Real World

```
packets can be\n```

```
easy to change implementation
```

```
compare them rather than name binding/bind/accept\n```

```
more abstract (much simpler):\n```

```
inter-packet time (msec) and packet size (byte)\n```

```
no addresses, just global variables\n```

```
--\n```

```
+ 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0\n```

```
r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0\n```

```
+ 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0\n```

```
r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0\n```

```
```
**Plumbing: Packet Flow**

**Computer Communications - CSCI 551**

Copyright © William C. Cheng

---

**TCP**

**Agent/TCP**

**Addr**

**Classifier**

**Port**

**Classifier**

**Application/FTP**

dst_=1.0

---

**Link n0-n1**

**entry_0**

**entry_1**

**entry_0**

**entry_1**

---

**Entry:**

**0**

**1**

**0**

**1**

---

**Set ns [new Simulator]**

**# Turn on tracing**

**Create topology**

**# Setup packet loss, link dynamics**

**# Create routing agents**

**# Create:**

**- multicast groups**

**- protocol agents**

**- application and/or setup traffic sources**

**# Post-processing proc**

**# Start simulation**

---

**Summary: Generic Script**

**Structure**

**Computer Communications - CSCI 551**

---

**Simple scenario with TCP and UDP connections**

**Example - TCP**

**TCP : Step 1**

**TCP : Step 2**

**Scheduler & Tracing**

**Example - TCP**

**TCP : Step 3**

**TCP**

**UDP**

**Simple scenario with TCP and UDP connections**

**TCP : Step 1**

**TCP : Step 2**

**TCP : Step 3**

---

**Structure**

**Summary: Generic Script**
TCP: Step 4

Create TCP agents

vp(n0 tcp)
vp(n3 sink)
ns attach-agent n0 tcp
ns attach-agent n3 sink
ns connect tcp sink

TCP: Step 5

Attaching traffic

set ftp [new Application/FTP]
ftp attach-agent tcp

# start application traffic
ns at 1.1 "ftp start"

TCP: Step 6

Packet-level animation

Nam-1 (Network Animator Version 1)

Well-supported by ns

Convert trace output into Xgraph format

Xgraph

Color mapping

Ns-nam Interface

Node manipulation

Link manipulation

Topology layout

Protocol state

Misc

Viz Tools

end of simulation wrapper (as usual)

Tcp : Step 6

Node manipulation

Link manipulation

Topology layout

Protocol state

Misc

Viz Tools

end of simulation wrapper (as usual)

Tcp : Step 6

Node manipulation

Link manipulation

Topology layout

Protocol state

Misc

Viz Tools

end of simulation wrapper (as usual)
Other NS Features

- Connect simulator in a real network
- Connect simulator in a real network
- Can receive and send out live packets from/to the real world
- GUI

Resources

- Ns distribution download
  - http://www.isi.edu/nsnam/ns/ns-build.html
- Installation problems and bug-fix
  - http://www.isi.edu/nsnam/ns/ns-problems.html
- Ns-users mailing list
  - Ns-users@isi.edu
- Ns-users archive
  - http://www.isi.edu/nsnam/ns/ns-users.html
- Marc Greis' tutorial
  - http://www.isi.edu/nsnam/ns/tutorial
- Practical programming in Tcl and Tk
  - Brent Welch

Resources (cont.)

- Ns-users archive
- Ns-manual
- http://www.isi.edu/nsnam/ns/nsdocumentation.html
- Tcl (Tool Command Language)
  - http://dev.scriptics.com/scripting
- Otcl (MIT Object Tcl)
  - ~otcl/doc/tutorial.html (in distribution)